Testing the functionality of `mtool`: Bias of `mtool` coefficient estimates

Nirupama Tamvada

Here we evaluate the ability of `mtool` to recover the true coefficients of the model covariates by looking at the bias of the estimated coefficients. We compare the bias of the recovered coefficients to that of `nnet` and `glmnet` , both of which allow for fitting a multinomial regression, while `glmnet` allows for fitting a penalized multinomial regression.

Regularized Multinomial Regression

When the categorical response variable G has $K > 2$ levels, the linear logistic regression model can be generalized to a multi-logit model.

`glmnet` uses a symmetric parameterization of the model:

$$\Pr(G = l|x) = \frac{e^{\beta_{0l}+x^T\beta_l}}{\Sigma_{k=1}^{K} e^{\beta_{0k}+x^T\beta_k}} \qquad (1)$$

This parameterization is well suited to a regularization setting as it is a less-than full-rank parameterization, and is not identifiable without constraints.

The traditional parameterization of the multinomial regression however, is to extend the logistic regression likelihood to $K-1$ logits

$$\log\frac{\Pr(G = l|x)}{\Pr(G = K|x)} = \beta_{0l} + x^T\beta_l, \quad l = 1, ..., K-1 \qquad (2)$$

`casebase` uses the (2) parameterization of the multinomial logistic model. With this parameterization, the coefficients are with respect to a reference category (the censored population in terms of our specific application) and the constant offset term is not dropped. Thus, the log-likelihood of the (2) parameterization must be penalized, and the `mtool` package was employed to this end.

Model Fitting Algorithm

The `mtool` implementation uses stochastic gradient descent, and thus is well-suited in terms of speed to problems with a large `N`.

We simulate data from a multinomial distribution with 3 classes (class 0 is the reference class). To return the correct results from `mtool`, it is very important to make sure that `Y` is numeric and that the categories are coded as 0, 1, 2, ...etc. with 0 as the reference.

Case 1: Non-sparse setting without regularization ($\alpha = 0$ and $\lambda = 0$)

1. Small N (N = 500)

```r
set.seed(200)
# Generate covariates
X <- matrix(rnorm(2500), 500, 5)

# coefficients for each choice
X1 <- rep(0, 5)
X2 <- rep(0.5, 5)
X3 <- rep(-1, 5)

# vector of probabilities
vProb = cbind(exp(X%*%X1), exp(X%*%X2), exp(X%*%X3))

# multinomial draws
mChoices <- t(apply(vProb, 1, rmultinom, n = 1, size = 1))
dfM <- cbind.data.frame(y = apply(mChoices, 1, function(x) which(x == 1)), X)
# Rename covariates
colnames(dfM)[2:6] <- paste0('x', 1:5)

# 0, 1, 2 for levels
Y <- factor(dfM$y-1)

# Covariate matrix
X <- as.matrix(dfM[, c(2:6)])

# Rename covariates
colnames(X) <- paste0('x', 1:5)
```

Fitting the three models:

```r
# nnet
fit.nnet <- nnet::multinom(y ~ x1 + x2 + x3 + x4 + x5 - 1,
                           data = dfM)

# glmnet
fit.glmnet <- fit.glmnet <- glmnet::glmnet(
  x = X, y = Y,
  family = "multinomial",
  intercept = FALSE,
  type.multinomial = "grouped",  # same sparsity pattern for all outcome classes
  lambda = 0)

# mtool
capture.output(
fit.mtool <-  mtool::mtool.MNlogistic(
  X = X,
  Y = as.numeric(Y),
  offset = rep(0, length(Y)),
  N_covariates = 0,
  regularization = 'l1l2',
  transpose = FALSE,
  lambda1 = 0)
)
```

Results table for coefficients for class 2:

Covariates	True coefficients	nnet	glmnet	mtool
x1	0.5	0.689	0.808	0.549
x2	0.5	0.583	0.766	0.558
x3	0.5	0.329	0.577	0.602
x4	0.5	0.491	0.722	0.592
x5	0.5	0.311	0.612	0.669

2. Large N (N = 5,000)

```r
set.seed(200)
# Generate covariates
X <- matrix(rnorm(25000), 5000, 5)

# coefficients for each choice
```

```r
x1 <- rep(0, 5)
x2 <- rep(0.5, 5)
x3 <- rep(-1, 5)

# vector of probabilities
vProb = cbind(exp(X%*%x1), exp(X%*%x2), exp(X%*%x3))

# multinomial draws
mChoices <- t(apply(vProb, 1, rmultinom, n = 1, size = 1))
dfM <- cbind.data.frame(y = apply(mChoices, 1, function(x) which(x == 1)), X)
# Rename covariates
colnames(dfM)[2:6] <- paste0('x', 1:5)

Y <- as.numeric(dfM$y-1)
X <- as.matrix(dfM[, c(2:6)])

# Rename covariates
colnames(X) <- paste0('x', 1:5)
```

Fitting the three models:

```r
# nnet
fit.nnet <- nnet::multinom(y ~ x1 + x2 + x3 + x4 + x5 - 1,
                           data = dfM)

# glmnet
fit.glmnet <- fit.glmnet <- glmnet::glmnet(
  x = X, y = Y,
  family = "multinomial",
  intercept = FALSE,
  type.multinomial = "grouped",  # same sparsity pattern for all outcome classes
  lambda = 0)

# mtool
fit.mtool <- mtool::mtool.MNlogistic(
  X = X,
  Y = Y, # I coded Y to start from 0, Y in (0,1,2) in your example
  offset = rep(0, length(Y)),
  N_covariates = 0,
  regularization = 'l1l2',
  transpose = FALSE,
```

```
  lambda1 = 0
)
```

Results table for coefficients for class 2:

Covariates	True coefficients	nnet	glmnet	mtool
x1	0.5	0.457	0.651	0.471
x2	0.5	0.56	0.696	0.573
x3	0.5	0.48	0.655	0.495
x4	0.5	0.553	0.683	0.567
x5	0.5	0.494	0.693	0.508

Case 2: Sparse settings with regularization ($\alpha = 0.5$, Elastic Net)

Elastic Net Penalization in `mtool`

The optimization performed on `mtool` calls the `Proximal Toolbox` , which is a toolbox in the larger `SPAMS` (SPArse Modeling Software) optimization toolbox for sparse decomposition problems. For the Elastic-Net penalty, for every column of u of $U = [u^1, ..., u^n]$ in $\mathbb{R}^{p \times n}$, one column of $V = [v^1, ..., v^n]$ in $\mathbb{R}^{p \times n}$ is computed to solve the following proximal operator:

$$\min_{v \in \mathbb{R}^p} \frac{1}{2}||u - v||_2^2 + \lambda_1||v||_1 + \lambda_2||v||_2^2$$

We set $\lambda_1 = \lambda\alpha$ and $\lambda_2 = \frac{\lambda(1-\alpha)}{2}$ to obtain the Elastic Net penalty implemented in `glmnet` as well from Zou and Hastie., 2005.

1. N > p (N = 1000, p = 10)

```
set.seed(200)
# Generate covariates
X <- matrix(rnorm(10000), 1000, 10)

# coefficients for each choice with some sparsity
X1 <- rep(0, 10)
X2 <- c(rep(0.5, 5), rep(0, 5))
zero_X2 <- which(X2 == 0)

X3 <- c(rep(-1, 4), rep(0, 6))
```

```
zero_X3 <- which(X3 == 0)


# vector of probabilities
vProb = cbind(exp(X%*%X1), exp(X%*%X2), exp(X%*%X3))

# multinomial draws
mChoices <- t(apply(vProb, 1, rmultinom, n = 1, size = 1))
dfM <- cbind.data.frame(y = apply(mChoices, 1, function(x) which(x == 1)), X)
# Rename covariates
colnames(dfM)[2:11] <- paste0('x', 1:10)

# 0, 1, 2 for levels
Y <- as.numeric(dfM$y-1)

# Covariate matrix
X <- as.matrix(dfM[, c(2:11)])

# Rename covariates
colnames(X) <- paste0('x', 1:10)
```

Fitting the two models (**nnet** does not provide a penalized implementation of the multinomial regression):

```
# glmnet
fit.glmnet <- fit.glmnet <- glmnet::glmnet(
  x = X, y = Y,
  family = "multinomial",
  intercept = FALSE,
  type.multinomial = "grouped",  # same sparsity pattern for all outcome classes
  lambda = 0.1, alpha = 0.5)

 # Elastic-net reparametrization
alpha <- 0.5
lambda <- 0.1
lambda1 <- lambda*alpha
lambda2 <- 0.5*lambda*(1 - alpha)
# mtool
fit.mtool <- mtool::mtool.MNlogistic(
  X = cbind(X),
  Y = Y,
```

```
  offset = rep(0, length(Y)),
  N_covariates = 0,
  regularization = 'elastic-net',
  transpose = FALSE,
  lambda1 = lambda1, lambda2 = lambda2,
            lambda3 = 0
)
```

Results table for coefficients for class 2:

Covariates	True coefficients	glmnet	mtool
x1	0.5	0.327	0.234
x2	0.5	0.371	0.308
x3	0.5	0.335	0.24
x4	0.5	0.398	0.295
x5	0.5	0.015	0
x1	0	0	0
x2	0	0	0
x3	0	0	0
x4	0	0	0
x5	0	0	0

Results table for coefficients for class 3:

Covariates	True coefficients	glmnet	mtool
x1	-1	-0.364	-0.306
x2	-1	-0.367	-0.29
x3	-1	-0.38	-0.333
x4	-1	-0.421	-0.339
x5	0	-0.005	0
x1	0	0	0
x2	0	0	0
x3	0	0	0
x4	0	0	0
x5	0	0	0

Number of non-zero coefficients:

2. p > N (N = 50, p = 100)

```r
set.seed(200)
# Generate covariates
X <- matrix(rnorm(500), 50, 100)

# coefficients for each choice with some sparsity
X1 <- rep(0, 10)
X2 <- c(rep(0.5, 5), rep(0, 5))
X3 <- c(rep(-1, 4), rep(0, 6))

# vector of probabilities
vProb = cbind(exp(X%*%X1), exp(X%*%X2), exp(X%*%X3))

# multinomial draws
mChoices <- t(apply(vProb, 1, rmultinom, n = 1, size = 1))
dfM <- cbind.data.frame(y = apply(mChoices, 1, function(x) which(x == 1)), X)
# Rename covariates
colnames(dfM)[2:11] <- paste0('x', 1:10)

# 0, 1, 2 for levels
Y <- as.numeric(dfM$y-1)

# Covariate matrix
X <- as.matrix(dfM[, c(2:11)])

# Rename covariates
colnames(X) <- paste0('x', 1:10)
```

Results table for coefficients for class 2: